Quick Incompact3d guide and notes

This is a quick guide on to how to use **Incompact3d** and to know some details of its functioning.

Compiling

It is suggested to create a *build* directory in the same folder of the solver.

```
mkdir build
```

In the *build* directory run

```
cmake ../
```

After that, the *makefile* will be created. It is now possibly to compile the binary file in the build directory with the following command:

```
make -j n
```

where `n` is the number of processors that will be used to compile the program.

Basic functioning

In this section, the main features of the code that can be useful to run simulations are reported.

In Incompact3d:

1. Dimensional Navier-Stokes equations are solved.
2. The Re specified in the input file is used to compute the kinematic viscosity as

$$\nu = \frac{1}{Re}$$

   so we are assuming unitary reference length and velocity scales. Reference scales depend then on the specific flow case.

   As an example, for a channel flow, the reference velocity is the bulk velocity U_B and the reference length is the channel half-height h. Moreover, if constant pressure gradient (CPG) option is enabled, the Reynolds number to be specified is the friction Reynolds number Re_τ. An approximate relation is available in order to estimate the related bulk Reynolds number:

$$Re_B \approx \frac{Re_\tau}{0.116}^{1/0.88}$$

   a similar relation is reported also by Pope:

$$Re_B \approx 0.09 Re_\tau^{0.88}$$

3. To evaluate the stretching parameter for the mesh β, some useful scripts can be found in the folder `1-pre_processing/Stretching mesh`. The default code of Incompact3d is `stretching_parameter_channel.f90` and it is used to setup channel flow simulations. For temporal TBL simulations, the Python script `mesh_evaluation_ttbl.py` was developed. Low β values correspond to a strong stretching of the mesh, while high β values correspond to almost uniform mesh. Pay attention

that strong stretching of the mesh causes the code to be much more unstable, in particular if noise or fluctuations are used to trigger transition at the beginning of simulations.

4. Variables are saved on y_p points, that are the faces of the grid elements. On the other hand, y_{pi} points are the centers of the grid elements (i : internal).
5. Boundary values of velocity are specified through the b_{ijk} variables, where i is the wall-normal direction of the boundary, j is the direction of the specific velocity component and k specifies if we are considering the bottom or the top walls (e.g. b_{yx1} refers to the x velocity component, specified at the bottom boundary with normal direction y).
6. Boundary conditions are specified in the input file `input.i3d` with the following variables: `nclx1, nclxn, ncly1, nclyn, nclz1, nclzn`, that specify the normal direction to the boundary and if we are considering the first or the last element along the specific direction. Values that can be adopted are: 0, for *periodic BC*, 1 for *free-slip BC* and 2 for *Dirichlet BC* (so imposed velocity value). Boundary conditions can be different along the same dimension, so different combinations can be enforced.
7. Boundary conditions for scalar fields have the same functioning of the velocity BCs. They are called: `nclxS1, nclxSn, nclyS1, nclySn, nclzS1, nclzSn`.